# Frameworks DBI para Seguridad Informática: usos y comparativa

Juan Antonio Artal, **Ricardo J. Rodríguez**, José Merseguer



jaartal@gmail.com, {rjrodriguez, jmerse}@unizar.es
tw: @RicardoJRdez – http://www.ricardojrodriguez.es


Universidad de Zaragoza
Zaragoza, Spain

3 de Noviembre, 2012

**No cON Name 2012**
Barcelona, España

# Hasta siempre, comandante!

## In memoriam

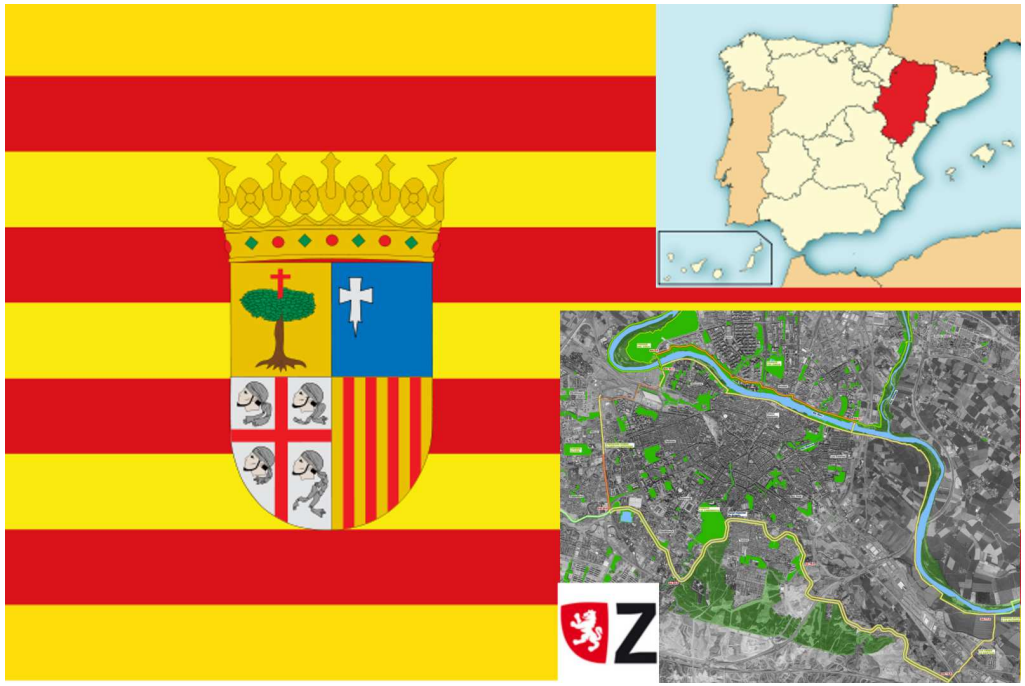
*"Y sabed que el ayer es sólo la memoria del hoy y el mañana es el ensueño del hoy"*

**El profeta** (Gibran Jalil Gibran)

# $whoami

Universidad Zaragoza
1542

# $whoami



Universidad Zaragoza
1542

# $whoami

Universidad Zaragoza
1542

# $whoami

Universidad Zaragoza
1542

# $whoami



Universidad Zaragoza
1542

# $whoami



- Miembro de CLS desde sus inicios (2000)
- Investigador (PhD candidate) en Universidad de Zaragoza
  - Rendimiento de sistemas software complejos
  - Ingeniería de Software segura
  - Sistemas de Tolerancia a Fallos (diseño y modelado)
  - Análisis *malware*

# Licencia de Código Desarrollado

- GPL v3
  (`http://gplv3.fsf.org/`)
- Intel Open Source License
  (`http://opensource.org/licenses/intel-open-source-license.html`)
- Especificado en cada fichero de código fuente



- Código descargable de:

`http://webdiis.unizar.es/~ricardo/files/NcN2k12.tar.gz`
(proyecto VS2008 + presentación PDF)


Universidad Zaragoza

# Outline


Universidad Zaragoza

# DBI: ¿qué es? (I)

**Instrumentación** ??
**Dinámica** de ??
**Ejecutables** ??


Universidad Zaragoza

# DBI: ¿qué es? (I)

**Instrumentación** ??
**Dinámica** de ??
**Ejecutables** ??


Universidad Zaragoza

# DBI: ¿qué es? (II)

## ¿Instrumentación?

### Instrumentación

- "Capacidad de observar, monitorizar y modificar el comportamiento de un programa" (Gal Diskin)
- Inserción de código adicional en un programa para recoger información


Universidad Zaragoza

# DBI: ¿qué es? (II)

## ¿Instrumentación?

### Instrumentación

- "Capacidad de observar, monitorizar y modificar el comportamiento de un programa" (Gal Diskin)
- Inserción de código adicional en un programa para recoger información
- Análisis y control sobre qué está pasando en el código de un programa
  - Recopilar información
  - Inserción de código (arbitrario)

Universidad Zaragoza
1542

# DBI: ¿qué es? (III)

**Instrumentación** ??
**Dinámica** de ??
**Ejecutables** ??

Universidad Zaragoza
1542

# DBI: ¿qué es? (III)

| | |
|---|---|
| **Instrumentación** | Qué está pasando... |
| **Dinámica** de | ?? |
| **Ejecutables** | ?? |

Universidad Zaragoza

# DBI: ¿qué es? (III)

| | |
|---|---|
| **Instrumentación** | Qué está pasando... |
| **Dinámica** de | ?? |
| **Ejecutables** | ?? |

Universidad Zaragoza

# DBI: ¿qué es? (IV)

## ¿Dinámica?

### Análisis de código

- Estático
  - ANTES de la ejecución
  - Todos los posibles caminos del programa
- Dinámico
  - DURANTE la ejecución
  - Un posible camino del programa (dependencia de datos de entrada)


Universidad Zaragoza

# DBI: ¿qué es? (V)

| | |
|---|---|
| **Instrumentación** | Qué está pasando... |
| **Dinámica** de | ?? |
| **Ejecutables** | ?? |


Universidad Zaragoza

# DBI: ¿qué es? (V)

| | |
|---|---|
| **Instrumentación** | Qué está pasando... |
| **Dinámica** de | DURANTE la ejecución... |
| **Ejecutables** | ?? |

Universidad Zaragoza

# DBI: ¿qué es? (V)

**Instrumentación**
**Dinámica** de
**Ejecutables**

Qué está pasando...
DURANTE la ejecución...
??

Universidad Zaragoza
1542

# DBI: ¿qué es? (IV)

## ¿Ejecutables?

### Análisis dinámico

- Código fuente disponible
  - Código fuente
  - Compilador
- Sin código fuente
  - Ejecutable
    - Estático (i.e., crea nuevo ejecutable modificado)
    - Dinámico
  - Entorno
    - Emulación
    - Virtualización
  - Debugging


Universidad Zaragoza

# DBI: ¿qué es? (VI)

| | |
|---|---|
| **Instrumentación** | Qué está pasando... |
| **Dinámica** de | DURANTE la ejecución... |
| **Ejecutables** | ?? |

Universidad Zaragoza

# DBI: ¿qué es? (VI)

**Instrumentación** Qué está pasando...
**Dinámica** de DURANTE la ejecución...
**Ejecutables** de un ejecutable (binario)...

Universidad Zaragoza

# DBI: ¿qué es? (VII)

## Ventajas de DBI

### Instrumentación de Ejecutables: ventajas

- Independiente de lenguaje de programación
- Visión modo máquina
- Instrumentación de software propietario

Universidad Zaragoza

# DBI: ¿qué es? (VII)

## Ventajas de DBI

### Instrumentación de Ejecutables: ventajas

- Independiente de lenguaje de programación
- Visión modo máquina
- Instrumentación de software propietario

### Instrumentación Dinámica: ventajas

- No se necesita recompilar/reenlazar cada vez
- Descubrir código *on-the-fly*
- Código generado dinámicamente
- Instrumentar proceso en ejecución (*attach*)

Zaragoza 1542

# DBI: ¿qué es? (IIX)

## Desventajas de DBI

### Principales desventajas

- Sobrecarga (instrumentación en tiempo de ejecución)
- $\Downarrow$ rendimiento (desesperación del analista!)

# DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa

**Código en ejecución**

Universidad Zaragoza 1542

# DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa
- ¿Qué inserto? → función de instrumentación

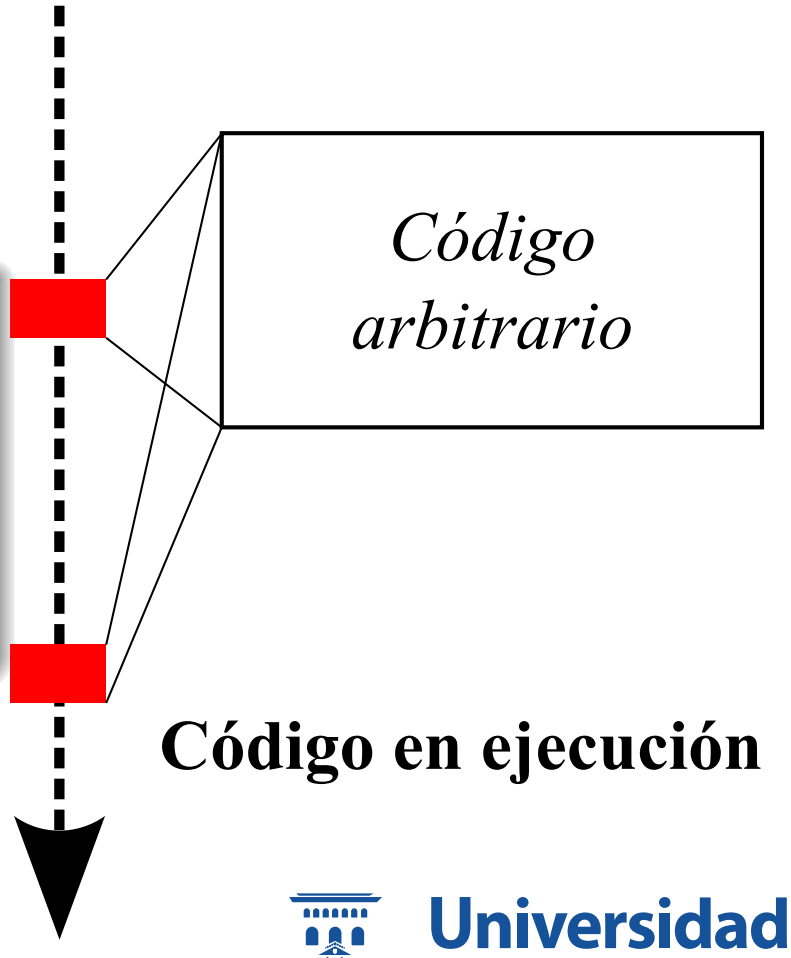
*Código arbitrario*

**Código en ejecución**

Universidad Zaragoza 1542

# DBI: ¿cómo funciona? (I)

- Inserción de código arbitrario durante ejecución de un programa
- ¿Qué inserto? → función de instrumentación
- ¿Dónde? → lugares de inserción



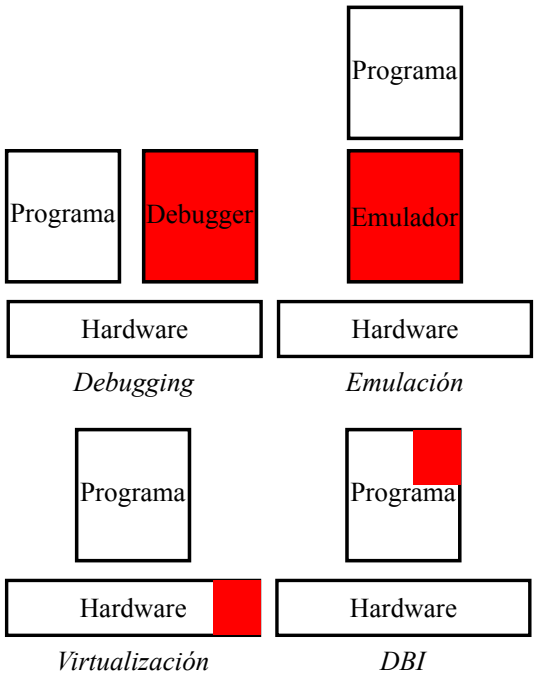

Universidad Zaragoza 1542

# DBI: ¿cómo funciona? (II)

## DBI en el contexto de análisis dinámico

### Definición (informal)

- Transformación del ejecutable
- Control total sobre la ejecución
- Sin necesidad de soporte arquitectural




- Virtualización
  - ¿Control total?
- Emulación
  - ¿Transformación del programa?
- Debugging
  - Soporte arquitectural

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria
  - Emulación de estrategias de especulación

# Usos de DBI (I)

## Usos no relacionados con seguridad

- Métricas y cobertura de código
- Generación de grafos de llamadas (*call-graphs*)
- Detección de fugas de memoria
- *Profiling* de instrucciones
- *Profiling* de dependencias de datos
- *Profiling* de threads
- Detección de condiciones de carrera
- Arquitectura de Computadores:
  - Generación de trazas
  - Modelado de predictores de saltos y caches
  - Recuperación de fallos de memoria
  - Emulación de estrategias de especulación
- . . .

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades


Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing

Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada

Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa

Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad


Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad
- Sandboxing


Universidad Zaragoza

# Usos de DBI (II)

## Usos relacionados con seguridad

- Análisis del flujo de datos (de control)
- Detección de vulnerabilidades
- Generación de casos de test / fuzzing
- Monitorización avanzada
- Ingeniería Inversa
- Monitorización de privacidad
- Sandboxing
- . . .


Universidad Zaragoza

# Usos de DBI (III)

## Algunas herramientas que usan DBI...

- Búsqueda de vulnerabilidades
  - SAGE (Microsoft)
  - Sogetis
  - Fuzzgrind
- Avalanche
- Determine
- Pincov
- Taintdroid
- VERA
- TraceSurfer
- ...
- TRIANA?? :)

# Usos de DBI (IV)

## Popularidad *in crescendo* (1)

- Covert Debugging: Circumventing Software Armoring, D. Quist & Valsmith, BH USA 2007/DefCon 15
- Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs (P. Bania, CoRR abs/0905.4581 2009)
- Tarte Tatin Tools: a set of plugins for malware analysis with Pin, (D. Reynaud, DeepSec 2009)
- Dynamic Binary Instrumentation for Deobfuscation and Unpacking (J-Y. Marion & D. Reynaud, DeepSec 2009)
- Dumping Shellcode with Pin (S. Porst, Zynamics 2010)
- Binary Instrumentation for Security Professionals (G. Diskin, BH USA 2011)
- Shellcode Analysis using Dynamic Binary Instrumentation (D. Radu & B. Dang, CARO 2011)

# Usos de DBI (V)

## Popularidad *in crescendo* (2)

- Hacking using Dynamic Binary Instrumentation (G. Diskin, HITB 2012 AMS)
- Improving Unpacking Process using DBI techniques (R.J. Rodríguez, RootedCON 2012)
- Improving Software Security with Dynamic Binary Instrumentation (R. Johnson, InfoSec Southwest 2012)
- Vulnerability Analysis and Practical Data Flow Analysis & Visualization (J.W. Oh, CanSecWest 2012)
- Light and Dark side of Code Instrumentation (D. Evdokimov, CONFidence 2012)
- Dynamic Binary Instrumentation Frameworks: I know you're there spying on me (F. Falcon & N. Riva, RECon 2012)

1542

# Outline


Universidad Zaragoza

# Framework DBI: ¿qué es? (I)

- APIs para desarrollo de herramientas
- DBA: Dynamic Binary Analysis tool
- Tipos de DBAs:
  - Ligeras
  - Pesadas (usa código intermedio)

# Framework DBI: ¿qué es? (I)

- APIs para desarrollo de herramientas
- DBA: Dynamic Binary Analysis tool
- Tipos de DBAs:
  - Ligeras
  - Pesadas (usa código intermedio)
- Componentes principales
  - Núcleo: compilador just-in-time (JIT)
    - Controla ejecución de aplicación
  - Librería (herramienta desarrollada)
    - ¿Dónde?
    - ¿Qué?

$ < *nucleo_fw_DBI* > < *herramienta/libreria* > < *ejecutable* >


Universidad Zaragoza

# Framework DBI: ¿qué es? (II)

## Modos de uso (comunes)

- JIT
  - Modificación de conjunto (pequeño) de ins. antes de ejecutarlas
  - Más robusto
  - Mejor para programas con comportamientos repetitivos (e.g., bucles)
- Probe
  - Parcheado en memoria
  - Menor sobrecarga (ejecución código nativo)
  - No soportado por todos los fws. DBI

Universidad Zaragoza
1542

# Framework DBI: ¿qué es? (II)

## Modos de uso (comunes)

- JIT
  - Modificación de conjunto (pequeño) de ins. antes de ejecutarlas
  - Más robusto
  - Mejor para programas con comportamientos repetitivos (e.g., bucles)
- Probe
  - Parcheado en memoria
  - Menor sobrecarga (ejecución código nativo)
  - No soportado por todos los fws. DBI

## Granularidad

| Instrucción | Bloque Básico | Superbloque | Traza | Rutina | Imagen |
|---|---|---|---|---|---|
| ++ | | | | | – – |

# Framework DBI: ¿qué es? (II)

## Modos de uso (comunes)

- JIT
  - Modificación de conjunto (pequeño) de ins. antes de ejecutarlas
  - Más robusto
  - Mejor para programas con comportamientos repetitivos (e.g., bucles)
- Probe
  - Parcheado en memoria
  - Menor sobrecarga (ejecución código nativo)
  - No soportado por todos los fws. DBI

## Granularidad

| Instrucción | Bloque Básico | Superbloque | Traza | Rutina | Imagen |
|---|---|---|---|---|---|
| ++ | | | | | – – |

→ No soportadas por todos los fws. . .

# Tipos de frameworks DBI (I)

## Fws DBI *on the wild*

| | | |
|---|---|---|
| Pin | DynInst | |
| Valgrind | Dtrace | HDtrans |
| DynamoRIO | Systemap | |

# Tipos de frameworks DBI (I)

## Fws DBI *on the wild*

| | | |
|---|---|---|
| Pin | DynInst | |
| Valgrind | Dtrace | HDtrans |
| DynamoRIO | Systemap | |

Mmm. . . ¿cuál es *más* mejor?

Universidad Zaragoza

# Tipos de frameworks DBI (I)

## Fws DBI *on the wild*

| | | |
|---|---|---|
| Pin | DynInst | |
| Valgrind | Dtrace | HDtrans |
| DynamoRIO | Systemap | |

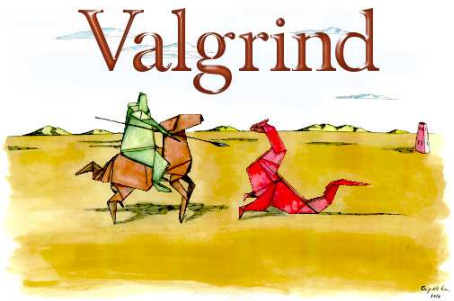
Mmm. . . ¿cuál es *más* mejor?

## Criterios de selección

- Software en desarrollo
- Licencia con acceso al fuente
- Gratuito
- API de desarrollo
- S.O. y arquitectura común

# Tipos de frameworks DBI (II)

## Diferencias y similitudes




### Características

- Tesis doctoral, Univ. Cambridge
- Código fuente disponible (GNU GPL v2)
- DBAs pesadas (código intermedio VEX IR)
- `http://www.valgrind.org`

Instrucciones Bloque básico Superbloque Traza Rutina IMagen

| Framework | Versión | Arquitecturas | S.O. | Granularidad |
|---|---|---|---|---|
| Valgrind | 3.8.1 (18/09/2012) | Arm, PowerPC, s390, x86, x64 | Android, OSX, Linux | I S |

# Tipos de frameworks DBI (II)

## Diferencias y similitudes


### Características

- Intel
- Código fuente disponible (Licencia propietaria)
- Vinculación a proceso en ejecucion
- `http://www.pintool.org/`

Instrucciones Bloque básico Superbloque Traza Rutina IMagen

| Framework | Versión | Arquitecturas | S.O. | Granularidad |
|---|---|---|---|---|
| Valgrind | 3.8.1 (18/09/2012) | Arm, PowerPC, s390, x86, x64 | Android, OSX, Linux | I S |
| Pin | 2.12 (10/10/2012) | Arm, IA-64, x86, x64 | Windows, Linux | I B T R M |

# Tipos de frameworks DBI (II)

## Diferencias y similitudes


### Características

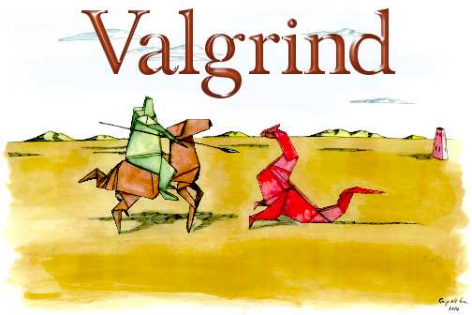
- MIT, HP, Google
- Código fuente disponible (BSD-2)
- Buena documentación
- `http://www.dynamorio.org/`

Instrucciones Bloque básico Superbloque Traza Rutina IMagen

| Framework | Versión | Arquitecturas | S.O. | Granularidad |
|---|---|---|---|---|
| Valgrind | 3.8.1 (18/09/2012) | Arm, PowerPC, s390, x86, x64 | Android, OSX, Linux | I S |
| Pin | 2.12 (10/10/2012) | Arm, IA-64, x86, x64 | Windows, Linux | I B T R M |
| DynamoRIO | 3.2.0-3 (01/03/2012) | x86, x64 | Windows, Linux | I B T |

# Tipos de frameworks DBI (II)

## Diferencias y similitudes






### Similitudes

- Código inyectado en C/C++
- No se necesita el código fuente de aplicación a instrumentar
- GNU/Linux x86

Instrucciones Bloque básico Superbloque Traza Rutina IMagen

| Framework | Versión | Arquitecturas | S.O. | Granularidad |
|---|---|---|---|---|
| Valgrind | 3.8.1 (18/09/2012) | Arm, PowerPC, s390, x86, x64 | Android, OSX, Linux | I S |
| Pin | 2.12 (10/10/2012) | Arm, IA-64, x86, x64 | Windows, Linux | I B T R M |
| DynamoRIO | 3.2.0-3 (01/03/2012) | x86, x64 | Windows, Linux | I B T |

# Comparativa entre Frameworks DBI (I)

## Herramientas DBA para comparativa

- Conteo de instrucciones
- Dos granularidades: instrucción y bloque básico

# Comparativa entre Frameworks DBI (I)

## Herramientas DBA para comparativa

- Conteo de instrucciones
- Dos granularidades: instrucción y bloque básico

## Objetivo de la comparativa

- Evaluar rendimiento de fws. DBI seleccionados
- *Slowdown*: $\frac{t_{instrumentado}}{t_{sin\_instrumentar}}$

Universidad Zaragoza 1542

# Comparativa entre Frameworks DBI (I)

## Herramientas DBA para comparativa

- Conteo de instrucciones
- Dos granularidades: instrucción y bloque básico

## Objetivo de la comparativa

- Evaluar rendimiento de fws. DBI seleccionados
- *Slowdown*: $\frac{t_{instrumentado}}{t_{sin\_instrumentar}}$

## Aprendizaje de las APIs

- Pin: ↑ Documentación, ↑↑ Ejemplos, ↑ Tutoriales
- DynamoRIO: ↑↑ Documentación, ↑ Ejemplos, ↑ Tutoriales
- Valgrind: ↓ Documentación, ↓ Ejemplos, ↓ Tutoriales

# Comparativa entre Frameworks DBI (II)

## Entorno de pruebas

- Hardware
  - Intel Core2 Duo 2GHz 667MHz, 2GiB DDR2, HDD 120GB
- Software
  - Fedora Core 14 32bits, gcc 4.5.1, GNU Fortran 4.5.1, r3


Universidad Zaragoza

# Comparativa entre Frameworks DBI (II)

## Entorno de pruebas

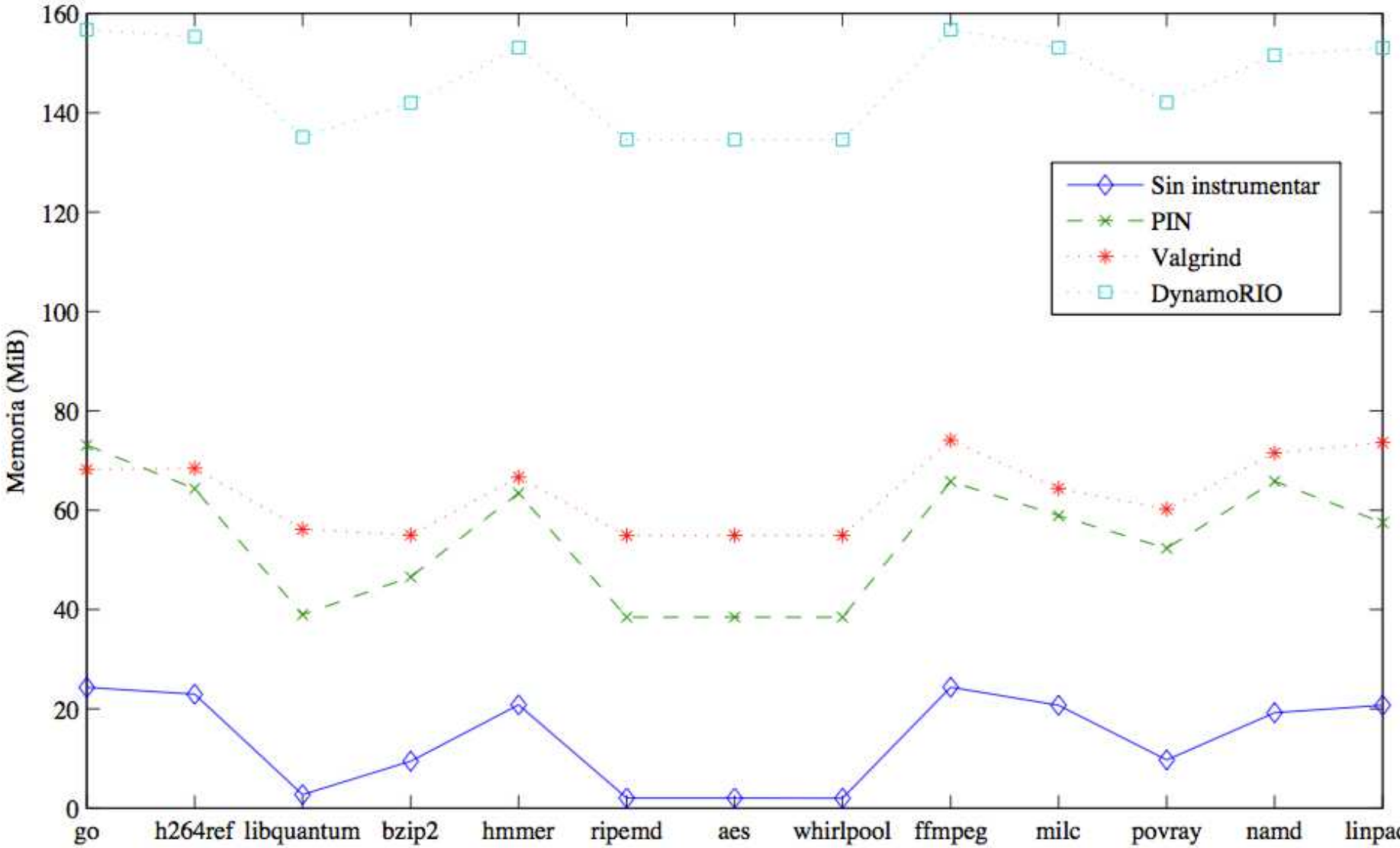
- Hardware
  - Intel Core2 Duo 2GHz 667MHz, 2GiB DDR2, HDD 120GB
- Software
  - Fedora Core 14 32bits, gcc 4.5.1, GNU Fortran 4.5.1, r3

## Benchmark

- Creación de benchmark propio para comparativa
- Alternativas estudiadas (e.g., SPEC) descartadas
- Diferentes categorías:
  - Cálculo entero
  - Cálculo real
  - E/S
  - Uso intensivo de memoria

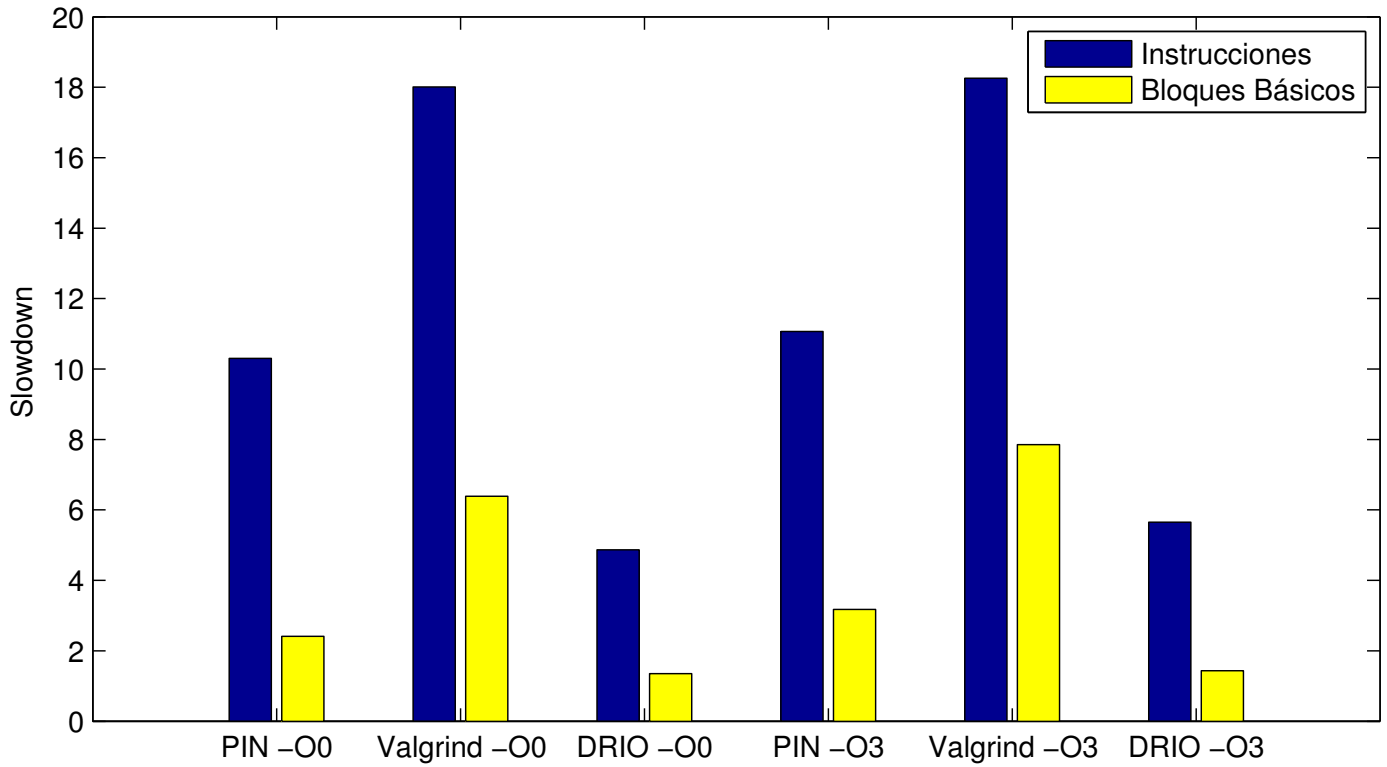
# Comparativa entre Frameworks DBI (III): Resultados (1)

## Consumo medio de memoria



# Comparativa entre Frameworks DBI (III): Resultados (2)

*Slowdown* por instrumentaciones




Universidad Zaragoza
1542

# Comparativa entre Frameworks DBI (III): Resultados (3)

## Conclusiones

- ✓ Nivel optimización o tipo cálculo → DynamoRIO
- X Más lenta → Valgrind
- Consumo de memoria
  - ✓ ↓ Pin
  - X ↑ DynamoRIO

## Cosas interesantes descubiertas...

- Conteo instrucciones difiere según fw. DBI → punto de inicio del núcleo DBI diferente
- Problema redondeo números 80 bits en máquinas de 32 ó 64 bits (Valgrind)
  - Ya reportado :( (`https://bugs.kde.org/show_bug.cgi?id=19791`)

# Comparativa entre Frameworks DBI (III): Resultados (4)

## Memoria del Proyecto

- Estudio comparativo de frameworks de Instrumentación Dinámica de Ejecutables (J.A. Artal)

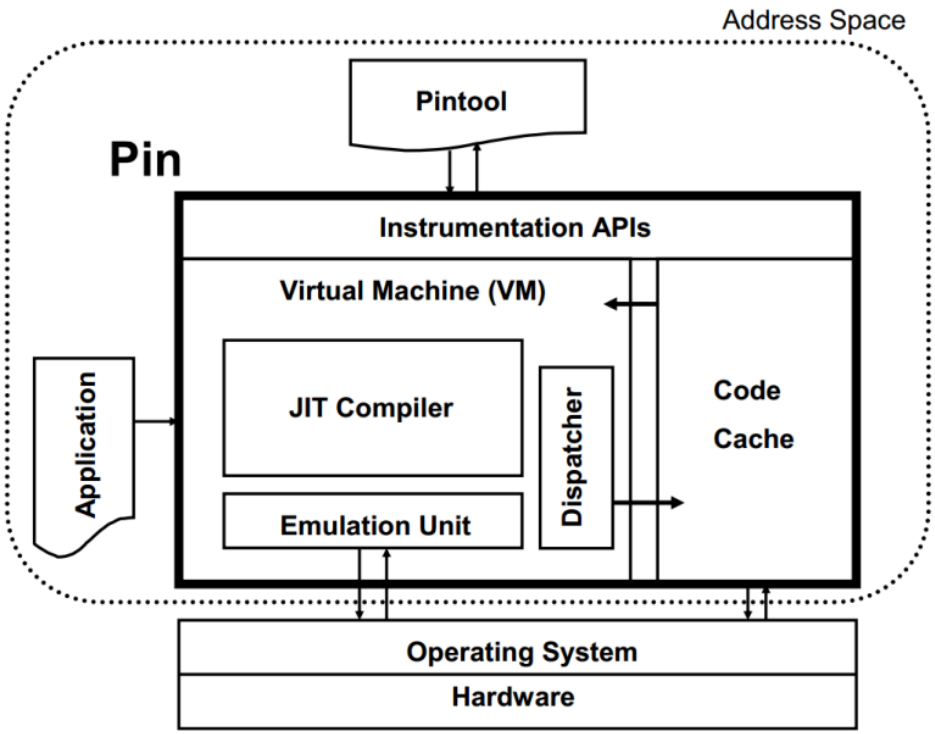
`http://webdiis.unizar.es/~ricardo/files/PFC.Estudio.Frameworks.DBI/Memoria_PFC_EstudioDBI.pdf`

Universidad Zaragoza 1542

# Outline


Universidad Zaragoza

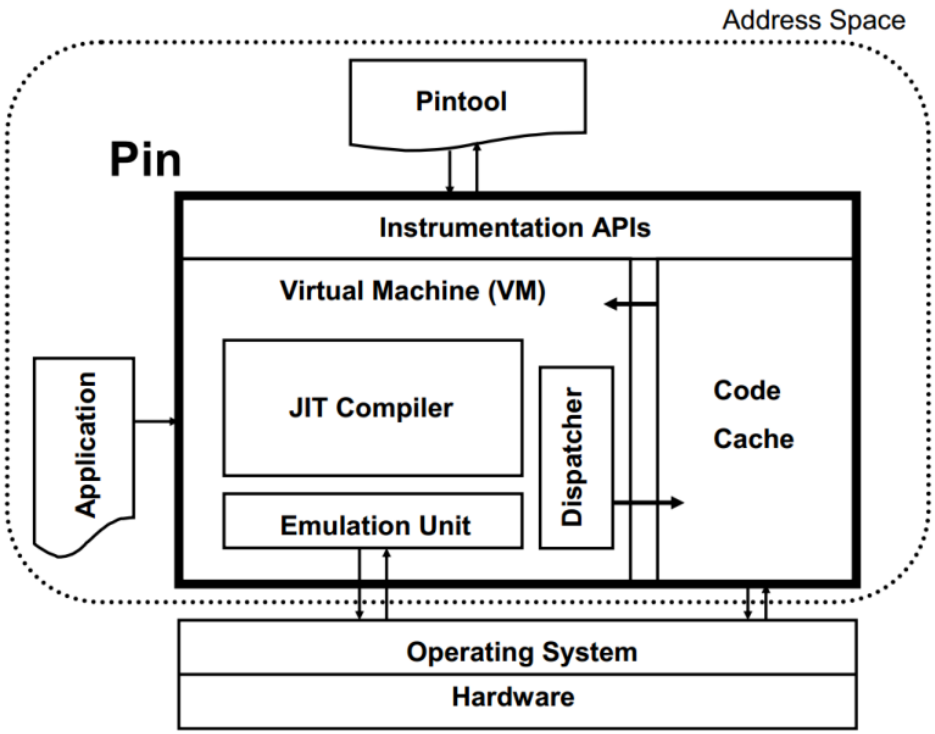
# Desarrollo de DBAs con Pin: Pintools (I)




- VM + cache de código + API instrumentación
- DBA → Pintool
- VM: JIT + emulador + dispatcher

Universidad Zaragoza
1542

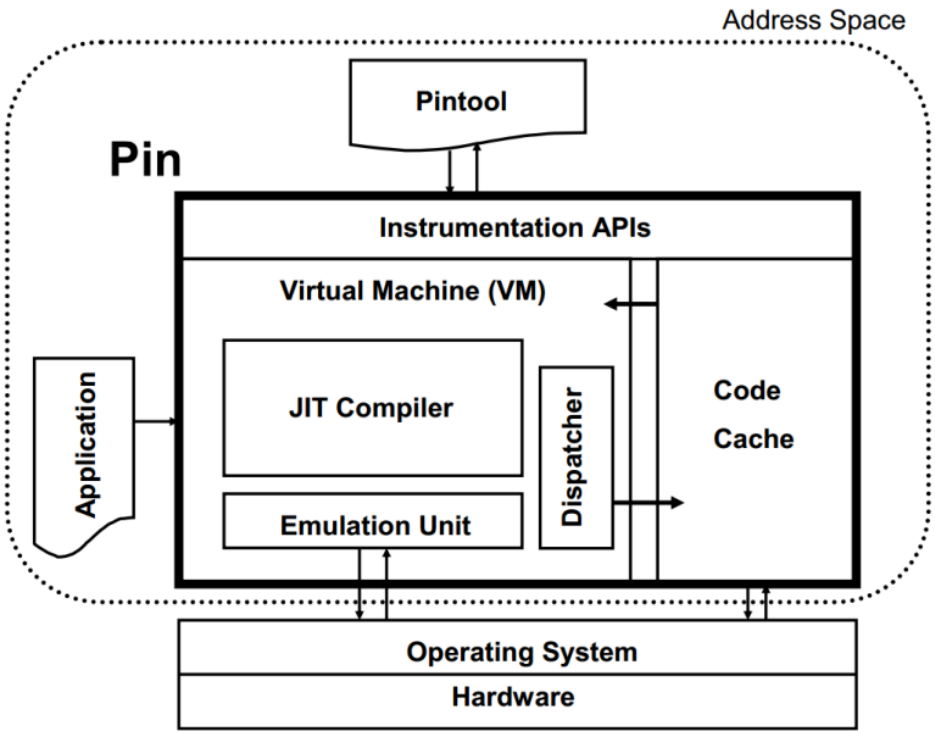
# Desarrollo de DBAs con Pin: Pintools (I)




- VM + cache de código + API instrumentación
- DBA → Pintool
- VM: JIT + emulador + dispatcher
  1. JIT compila e instrumenta código aplicación
  2. Lanzado por el dispatcher
  3. Guardado en cache de código

Universidad Zaragoza 1542

# Desarrollo de DBAs con Pin: Pintools (I)




- VM + cache de código + API instrumentación
- DBA → Pintool
- VM: JIT + emulador + dispatcher
  1. JIT compila e instrumenta código aplicación
  2. Lanzado por el dispatcher
  3. Guardado en cache de código
- Opera sobre S.O.: *user-space*

Universidad Zaragoza 1542

# Desarrollo de DBAs con Pin: Pintools (II)

## Ejemplo de desarrollo: `inscount.cpp`

```
#include "pin.H"

//Instruction counter
static UINT64 icount = 0;

// Called before every instruction is executed
VOID docount() { icount++; }

// Called every time a new instruction is encountered
VOID Instruction(INS ins, VOID *v){
    // Insert a call to docount before every instruction, no arguments are passed
    INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)docount, IARG_END);
}

// Called when the application exits
VOID Fini(INT32 code, VOID *v){
    std::cout << "Count " << icount << endl;
}

int main(int argc, char * argv[]){
    PIN_Init(argc, argv);
    INS_AddInstrumentFunction(Instruction, 0);
    PIN_AddFiniFunction(Fini, 0);
    PIN_StartProgram(); // no returns
    return 0;
}
```

rsidad
za

# Búsqueda de vulnerabilidades (I): Double Free

Demo: `DoubleFreeDBA.dll`

## Vulnerabilidad

- CWE-415 (`http://cwe.mitre.org/data/definitions/415.html`)
- Llamada a `free()` con la misma @ → memoria corrupta
- *"Doubly freeing memory may result in a write-what-where condition, allowing an attacker to execute arbitrary code"*

## DBA desarrollada con Pin (`DoubleFreeDBA.dll`)

- ¿Dónde?
  - APIs `RtlAllocateHeap` (después), `RtlAllocateFree` (antes)
- ¿Qué?
  - `RtlAllocateHeap`: guarda @ devuelta en lista
  - `RtlAllocateFree`: quita @ de lista, avisa si no está!

# Búsqueda de vulnerabilidades (I): Double Free

Demo: `DoubleFreeDBA.dll`

## Vulnerabilidad

- CWE-415 (`http://cwe.mitre.org/data/definitions/415.html`)
- Llamada a `free()` con la misma @ → memoria corrupta
- *"Doubly freeing memory may result in a write-what-where condition, allowing an attacker to execute arbitrary code"*

## DBA desarrollada con Pin (`DoubleFreeDBA.dll`)

- ¿Dónde?
  - APIs `RtlAllocateHeap` (después), `RtlAllocateFree` (antes)
- ¿Qué?
  - `RtlAllocateHeap`: guarda @ devuelta en lista
  - `RtlAllocateFree`: quita @ de lista, avisa si no está!

**Recuerda:** Hacer demo...

# Búsqueda de vulnerabilidades (II): Buffer Overflow (1)

Demo: `BufferOverflowDBA.dll`

## Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

## DBA desarrollada con Pin (`BufferOverflowDBA.dll`)

- Centrado en `scanf`
- ¿Dónde?→ API `scanf` (antes)
- ¿Qué?
  - Chequea parámetros buscando buffers sin acotar
- Mejoras: otras funciones vulnerables (e.g., `strcpy`)

# Búsqueda de vulnerabilidades (II): Buffer Overflow (1)

Demo: `BufferOverflowDBA.dll`

## Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

## DBA desarrollada con Pin (`BufferOverflowDBA.dll`)

- Centrado en `scanf`
- ¿Dónde?→ API `scanf` (antes)
- ¿Qué?
  - Chequea parámetros buscando buffers sin acotar
- Mejoras: otras funciones vulnerables (e.g., `strcpy`)

**Recuerda:** Hacer demo...

# Búsqueda de vulnerabilidades (II): Buffer Overflow (2)

Demo: `ProtectRetAddrDBA.dll`

## Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

## DBA desarrollada con Pin (`ProtectRetAddrDBA.dll`)

- ¿Dónde? → toda `CALL` (antes) o `RETN` (antes) en la sección `.text`
- ¿Qué?
  - `CALL` → guarda dirección legítima de retorno ($EIP + size(CALL)$)
  - `RETN` → si no está en la lista raro...
- Detectados 6 cambios de en librería `ntdll.dll`!!


Zaragoza

# Búsqueda de vulnerabilidades (II): Buffer Overflow (2)

Demo: `ProtectRetAddrDBA.dll`

## Vulnerabilidad

- CWE-120 (`http://cwe.mitre.org/data/definitions/120.html`)
- Copia a buffers sin restricciones → ejecución de código arbitrario
- *"Buffer overflows often can be used to execute arbitrary code [...]. Buffer overflows generally lead to crashes [...]."*

## DBA desarrollada con Pin (`ProtectRetAddrDBA.dll`)

- ¿Dónde? → toda `CALL` (antes) o `RETN` (antes) en la sección `.text`
- ¿Qué?
  - `CALL` → guarda dirección legítima de retorno ($EIP + size(CALL)$)
  - `RETN` → si no está en la lista raro...
- Detectados 6 cambios de en librería `ntdll.dll`!!

**Recuerda:** Hacer demo...

# Búsqueda de vulnerabilidades (III): Análisis *taint*

Demo: `TaintAnalysisDBA.dll`

## DBA desarrollada con Pin (`TaintAnalysisDBA.dll`)

- Análisis *taint* de parámetro `scanf`
- *Hook* en llamada de comparación de cadena `lstrcmpA`
- ¿Dónde? → API `scanf` (después)
- ¿Qué?
  - Tracea los registros/memoria contaminados a partir de la entrada

Universidad Zaragoza 1542

# Búsqueda de vulnerabilidades (III): Análisis *taint*

Demo: `TaintAnalysisDBA.dll`

## DBA desarrollada con Pin (`TaintAnalysisDBA.dll`)

- Análisis *taint* de parámetro `scanf`
- *Hook* en llamada de comparación de cadena `lstrcmpA`
- ¿Dónde? → API `scanf` (después)
- ¿Qué?
  - Tracea los registros/memoria contaminados a partir de la entrada

**Recuerda:** Hacer demo...

Universidad Zaragoza 1542

# Búsqueda de vulnerabilidades (IV): Ingeniería Inversa

Demo: `EasyPasswordDBA.dll`

## DBA desarrollada con Pin (`EasyPasswordDBA.dll`)

- Búsqueda de clave correcta
- *Hook* en llamada de comparación de cadena `lstrcmpA`
- ¿Dónde?
  - API `lstrcmpA` (antes)
- ¿Qué?
  - Guarda los parámetros de la función

Universidad Zaragoza

# Búsqueda de vulnerabilidades (IV): Ingeniería Inversa

Demo: `EasyPasswordDBA.dll`

## DBA desarrollada con Pin (`EasyPasswordDBA.dll`)

- Búsqueda de clave correcta
- *Hook* en llamada de comparación de cadena `lstrcmpA`
- ¿Dónde?
  - API `lstrcmpA` (antes)
- ¿Qué?
  - Guarda los parámetros de la función
- Esto ya no vale para las aplicaciones de verdad... ¿o sí? }:)


Universidad Zaragoza

# Búsqueda de vulnerabilidades (IV): Ingeniería Inversa

Demo: `EasyPasswordDBA.dll`

## DBA desarrollada con Pin (`EasyPasswordDBA.dll`)

- Búsqueda de clave correcta
- *Hook* en llamada de comparación de cadena `lstrcmpA`
- ¿Dónde?
  - API `lstrcmpA` (antes)
- ¿Qué?
  - Guarda los parámetros de la función
- Esto ya no vale para las aplicaciones de verdad... ¿o sí? }:)

**Recuerda:** Hacer demo...


Universidad Zaragoza

# Outline


Universidad Zaragoza

# Conclusiones

- Frameworks DBI: rápido y sencillo → alto potencial
- NO es necesario conocimientos de programación S.O. avanzados
  - Podemos centrarnos en la "miga": la DBA
- Desventajas:
  - Conocimiento de la API
  - Tiempo de ejecución

## Sobre la comparativa...

- ✓ Nivel optimización o tipo cálculo → DynamoRIO
- X Más lenta → Valgrind
- Consumo de memoria
  - ✓ ↓ Pin
  - X ↑ DynamoRIO

1542

# Agradecimientos

Buena gente a la que dar las gracias. . .


- Gal Diskin
- Dimitry “D1g1” Evdokimov
- Francisco Falcon & Nahuel Riva
- CrackLatinoS (CLS)
- Asociación NoConName, mil gracias!


Universidad Zaragoza
1542

# Agradecimientos

Buena gente a la que dar las gracias. . .

- Gal Diskin
- Dimitry "D1g1" Evdokimov
- Francisco Falcon & Nahuel Riva
- CrackLatinoS (CLS)
- Asociación NoConName, mil gracias!
- A vosotros por aguantar este tostón. . .


Universidad Zaragoza

# Frameworks DBI para Seguridad Informática: usos y comparativa

Juan Antonio Artal, **Ricardo J. Rodríguez**, José Merseguer



jaartal@gmail.com, {rjrodriguez, jmerse}@unizar.es
tw: @RicardoJRdez – http://www.ricardojrodriguez.es


Universidad de Zaragoza
Zaragoza, Spain

3 de Noviembre, 2012

**No cON Name 2012**
Barcelona, España